

Archive Web Sites Using AJAX and GWT

Trey Roby

IPAC, California Institute of Technology, Pasadena, CA, USA

Abstract. The last three years have seen much change in web technology and have created some significant breakthroughs. We are now able to let the user interact with an archive from the Web browser in ways we have never thought possible. The Web browser is no longer a glorified batch processing terminal, but an interactive environment that allows the user to have a similar experience as one might expect with an installed desktop application. We can now provide web based FITS viewing and interaction without any plugins. Much of this is made possible using AJAX. AJAX technology has made an major impact on how we think about developing on the Web. Users expect more and are drawn to more interactive and intuitive web sites. The problem with the Javascript part of AJAX is that it does not scale well to large Web applications, is hard to debug, and a lot of browser specific code is required. Google Web Toolkit (GWT) provides the solution to this problem. With GWT, you write code in Java that is compiled into Javascript. GWT handles many of the browser-specific issues and provides you an environment to develop very powerful web sites. This talk will discuss the concepts behind AJAX and GWT. We will also show how using these technologies in an archive web site will create a truly interactive experience.

1 Introduction

In the past five years, it has become very clear that the Web and web-development are revolutionizing. We are seeing applications come to the web that we never imagined before. Many web applications were once thought to be limited to the desktop. Other web applications are using levels of networking never considered before. The Web is certainly becoming more central in peoples lives. The Web browser is the key application to have on a computer. This provides us with a great opportunity as developers to create Astronomy archive web sites that increasingly allow users to interact with and understand the data that they are searching for.

2 Very Brief History of Computing

To really understand this change, it is helpful to ask the question, ‘How did Microsoft make it big?’ The answer is found in how business worked before the PC. Everything was batch. The user used a batch mainframe terminal. He filled out a form, hit enter, and waited for the results. Microsoft brought interactive programming into a mainframe world. The way business used computers changed significantly.

The Web in the 90s and early 2000s was very similar to this mainframe batch model. The user filled out a form, hit submit, and waited for the results:

except this time he did it with a Web browser. Computing had returned to the batch model.

This approach was shattered on Feb 8, 2005 when Google introduced Google Maps. For the first time we saw a Web browser with an interactive application. A new standard was set for web development. AJAX web development began.

Now we are seeing AJAX-based web applications appearing everywhere. Google has introduced several more such as Calendar, Word processor and the very popular Gmail. Yahoo has followed suit and revamped their whole web site to be AJAX-based. They have some very powerful finance tools. The news websites are using this approach to some degree. The social media and entertainment sites are making heavy use of AJAX. There are all sorts of speciality sites such as doodle.com or rememberthemilk.com that are good examples of this new development approach. Everybody is being affected in some way. Web sites that are not using AJAX elements are looking very dated.

3 Technologies Behind the Revolution

There are several technologies behind this Web development revolution. The most obvious is AJAX. AJAX is really a acronym (Asynchronous JavaScript And XML) that represents a set of existing components that are being used together in new ways. The major technologies are:

- JavaScript
- Dynamic HTML
- Asynchronous Calls
- CSS
- DOM
- XML
- JASON

Lets look at some of these in more detail.

Dynamic HTML technology allows any part of the a web page to change without reloading the page. The HTML document is viewed as a tree that can be modified using JavaScript (see figure 1) . This view of the documents is called the Document Object Model (DOM). When the DOM is modified the page changes immediately. This allows the page to change according to what the user is doing.

The maturing of JavaScript is probably the most important of the changes we are seeing. JavaScript is a very powerful, flexible, C-like scripting language that runs natively in the browser. It is event driven. This means that anytime the user clicks or moves his mouse over or types on any part of the web page, a JavaScript function can be called. It also has access to the DOM so it can change some or all of the web page on any event.

Another key to JavaScript is that it can make asynchronous calls to the server. This is not the traditional page load call. JavaScript can call the server at anytime, get the results, and can change any part of the web page. This allows for updates to the web page without the user even knowing that the the server is being called. The AJAX approach works with many, small, quick calls to the server and updates part of the web page instead of doing large page reloads.

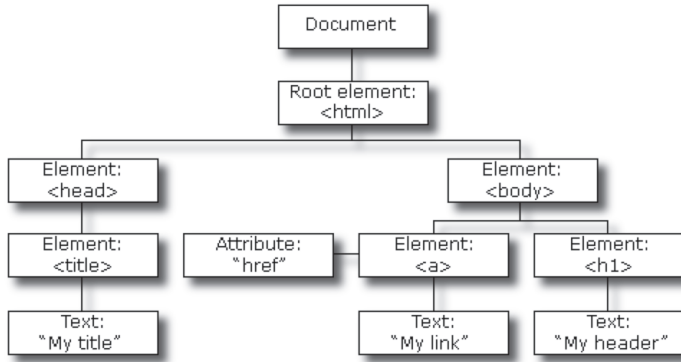


Figure 1. Document Object Model (DOM) example

The following two tables shows the difference between AJAX-based pages and traditional web pages.

- Classic Web
 - One server call = Whole Page Load
 - Server call = UI Generation
 - Server call = Large Overhead
 - Server call required for useful results
- AJAX Web
 - One Server call = Page update
 - One Server call = data only
 - One Server call = small overhead
 - Page can generate results without server call

Figure 2 shows the flow of a traditional web application while Figure 3 shows the flow of a AJAX web application.

All this technology working together allows the browser to now become an application environment. We can run application code on the browser allowing us to take advantage of client computing power and offloading the server. We can now once again start developing heavy clients. This gives the user a better, faster, more interactive experience. It takes advantage of the browsers full potential and creates opportunity to develop more powerful applications.

4 Challenges in Using AJAX

Doing AJAX development has significant challenges. Browsers can work differently. The JavaScript API can be slightly different. What will work in one browser may not work in another. There are many, many subtleties which require constant checking of the browser type and code that is full of ‘if’s’ just to do simple things.

JavaScript also has some significant weaknesses. It is not strongly typed like C or Java. It has a weak debugger and development environment support.

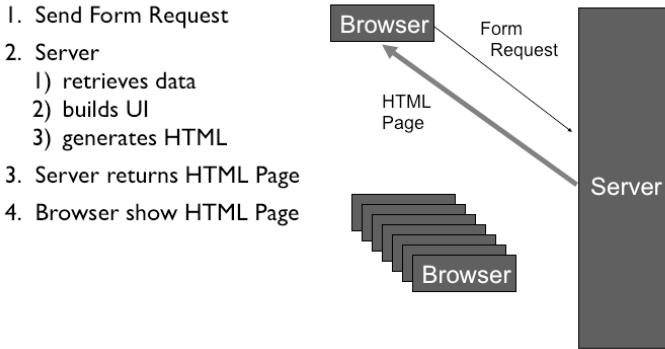


Figure 2. Traditional web app.

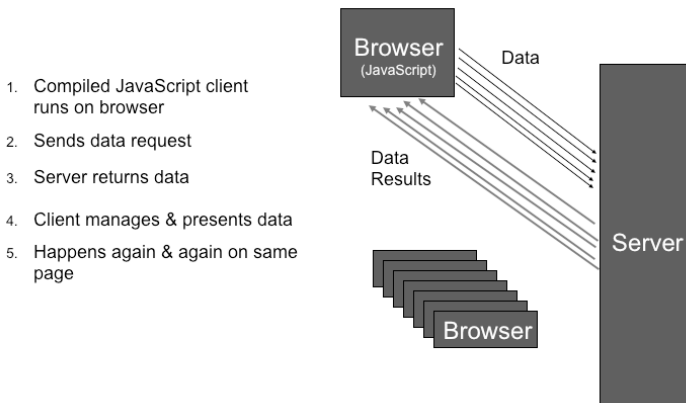


Figure 3. AJAX/javascript based web app.

It is not a compiled language. Syntax errors are not caught until runtime. It also means that it cannot be optimized until runtime. JavaScript is designed to be a scripting language so it has the standard scripting languages weaknesses, such as not working well in large applications. It has particular issues when more than one programmer is working on a project.

AJAX development provides us with great opportunity. We can write stunning web applications that will be a great benefit to the user and be easy to use. It also has non-trivial obstacles. It is hard to debug and requires lots of testing on browsers. Some think that you must be an 'AJAX guru' to even attempt this sort of development. It is also easy to write bad JavaScript code.

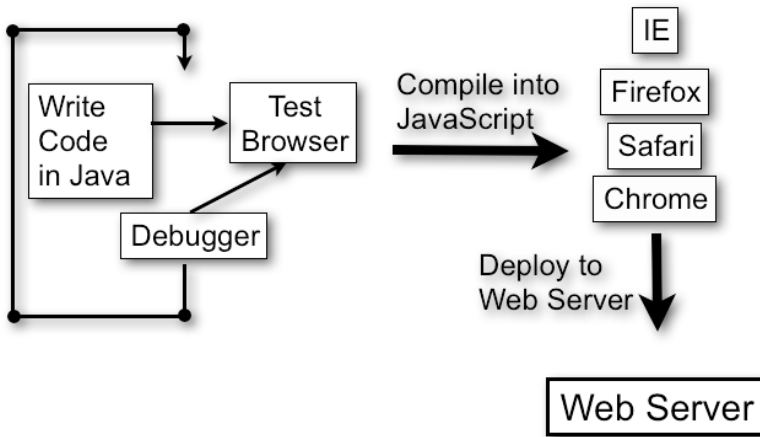


Figure 4. How to develop in GWT

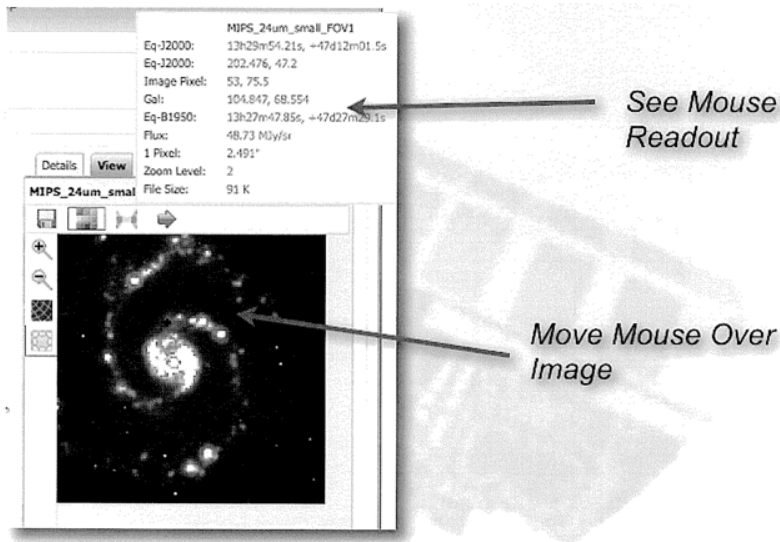


Figure 5. Image visualization example

5 Solution: Google Web Toolkit

The best solution we have seen for dealing with the challenges of AJAX development is using the Google Web Toolkit (GWT). GWT is a well supported, free, open source product that attempts to fix many of the AJAX challenges.

Using GWT, the developer writes code in Java instead of JavaScript. GWT compiles the Java into JavaScript making JavaScript into sort of a web browser

assembly language. This compilation creates all sorts of advantages. The biggest is that it does not just generate one set of JavaScript output. Instead, it creates a set of JavaScript output per browser type. Therefore, it can deal with most of the browser differences at compile time instead of runtime. It also compiles only what you actually use. For example, if you have a set of utility functions such as a math library, GWT will only bring in those functions that are actually used in the code. This way code is not downloaded to the browser that is not used, saving bandwidth. No browser plugin is required to make this happen.

GWT also provides several other features that are all optional to use. It has optional UI support so you can develop in a way similar to Java Swing. You can also build the UI by accessing the DOM directly. It provides an RPC environment for simplifying making asynchronous calls to the server. It is also possible to put bits of JavaScript directly into your GWT code. GWT allows you to do anything you can do in JavaScript and provides additional features.

Using this sort of development, we see many benefits. GWT handles many cross browser issues. We get the benefits of compilation and producing more optimal code. Java works better for large applications and has good debugger support. GWT comes with a development environment and test browser that interacts with your favorite Java debugger.

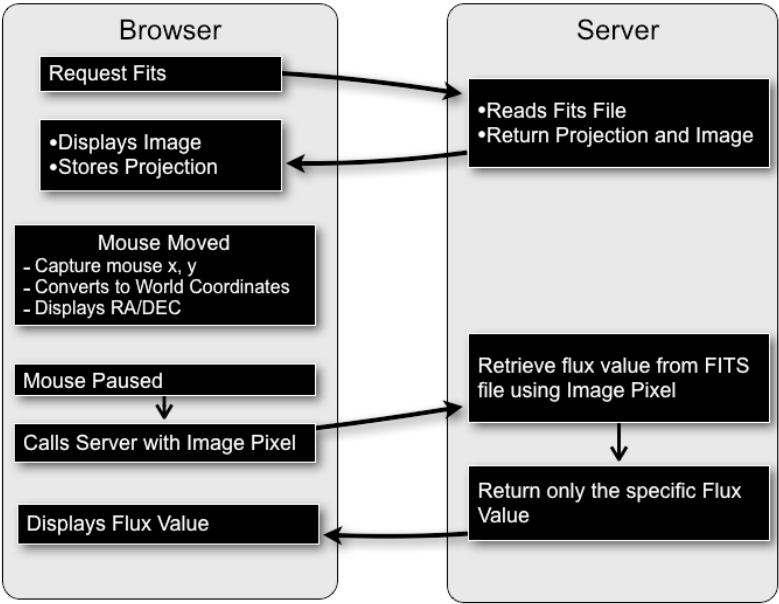


Figure 6. Data flow between browser and server

Figure 4 illustrates how the process of GWT development progresses. The developer starts writing in Java. Using GWT tools, the developer runs his code in a test browser. He can attach his favorite debugger to it if he desires. This iterative process continues until the developer is satisfied. When ready, the code

is compiled into JavaScript. One compilation per browser type. Finally the code is deployed onto a web server.

6 AJAX Example: Spitzer Heritage Archive Visualization

Lets look at a brief example from the Spitzer Heritage Archive. The Archive will show the user FITS image visualization (Figure 5). As the user moves his mouse, the information about where the mouse is on the image is updated. One field (flux) requires a server call and is only done when the mouse is paused. Figure 6 shows the AJAX calls, user interaction, and the dataflow as the user request a FITS image and then moves his mouse.

7 Conclusion

We can see that, while writing AJAX web applications has its challenges, it also has the greater rewards of better quality, more power, and certainly more user friendly applications. We have found using GWT is key to tapping into the power of this sort of development. It is a very effective approach to large scale development.

You are welcome to visit the Spitzer Heritage Archive at:

<http://shaweb1.ipac.caltech.edu/heritage/Heritage.html>